Available online at www.sciencedirect.com

ScienceDirect

Computer Speech & Language 46 (2017) 252–267

www.elsevier.com/locate/csl

Multilingually trained bottleneck features in spoken language recognition☆

Radek Fér*, Pavel Matějka, František Grézl, Oldřich Plchot, Karel Veselý, Jan Honza Černocký

Brno University of Technology, Speech@FIT and IT4I Center of Excellence, Božetěchova 2, Brno 612 66, Czech Republic

Received 5 October 2016; received in revised form 21 June 2017; accepted 28 June 2017
Available online 5 July 2017

Abstract

Multilingual training of neural networks has proven to be simple yet effective way to deal with multilingual training corpora. It allows to use several resources to jointly train a language independent representation of features, which can be encoded into low-dimensional feature set by embedding narrow bottleneck layer to the network. In this paper, we analyze such features on the task of spoken language recognition (SLR), focusing on practical aspects of training bottleneck networks and analyzing their integration in SLR. By comparing properties of mono and multilingual features we show the suitability of multilingual training for SLR. The state-of-the-art performance of these features is demonstrated on the NIST LRE09 database.

Keywords: Multilingual training; Bottleneck features; Spoken language recognition

1. Introduction

Neural networks (NN) have become a widely used technique for state-of-the-art Large Vocabulary Continuous Speech Recognition (LVCSR) systems and are rapidly expanding to other fields of speech recognition. Notably, bottleneck (BN) features (Kramer, 1991), extracted from a narrow layer of NN, have brought speech signal parametrization to a quantitatively different level (Grézl et al., 2007). These features convey information about phonetic content in a nonlinearly compressed form which can be directly used for the task of spoken language recognition (SLR), where they have demonstrated state of the art performance (Matějka et al., 2014; Jiang et al., 2014a; Ferrer et al., 2016).

Despite the excellent results, these features exhibit strong coupling to a language used during the NN training. This can be circumvented by means of multilingual training (Schultz and Waibel, 2001; Scanzio et al., 2008) and that is also the main focus of this paper. The term multilingual means that the NN is trained on several languages simultaneously. The NN thus learns (to some extent) a language independent representation of speech that gets

☆ This paper has been recommended for acceptance by Roger Moore.

* Corresponding author.
E-mail addresses: ifer@fit.vutbr.cz, radomilec@gmail.com (R. Fér).

http://dx.doi.org/10.1016/j.csl.2017.06.008
0885-2308/

encoded into bottleneck features. Such features were used for the LVCSR task and they were found to be superior to the ones trained on a single language (Scanzio et al., 2008; Veselý et al., 2012; Grézl et al., 2014).

In this paper, we extend our previous work (Fér et al., 2015) by showing more detailed results and further analysis of multilingual bottleneck features. Specifically, we focus on differences in mono- and multi-lingual features that should be addressed in context of SLR. We add experiments with different NN architectures and output layer setup. The SLR metrics are reported together with ASR related measures to see the level of their correlation. Note that all experiments from the original paper were re-scored using different SLR backend, so the corresponding values will not be the same.

The approach is tested on (clean) NIST LRE 2009 database (NIST, 2009), comparing the performance of monolingual (i.e., trained on single language) and multilingual systems. The results obtained with widely used Mel-frequency Cepstrum Coefficients (MFCC) Shifted Delta Cepstra (SDC) features (Torres-Carrasquillo et al., 2002) are included for reference.

1.1. Related work

Several different approaches to allow use of multilingual training corpora in SLR has been proposed. Zissman and Singer (1994) used six phoneme recognizers running in parallel, each producing a language-dependent likelihood based on an N-gram phonotactic model. Final score was obtained by averaging corresponding log-likelihoods. This is known as Parallel Phone Recognition followed by Language Modeling (PPRLM). Corredor-Ardoy et al. (1997) reported similar error rates to PPRLM approach when language dependent phonotactic models were trained on a merged phoneme set of four languages. The merging was done using Agglomerative Hierarchical Clustering with phoneme similarity based on Hidden Markov Model (HMM) phone likelihoods.

Big effort has been carried out for multilingual resource collection. Namely GlobalPhone, a high-quality multilingual database, was developed by the team from University of Karlsruhe (Schultz, 2002). In Schultz and Waibel (2001), International Phonetic Alphabet (IPA) was used to create a cross-lingual phoneme set by unifying the phoneme sets of different languages from this database.

The need for explicit phoneme set unification was mitigated in Scanzio et al. (2008) by dividing the output softmax layer of a NN into a set of independent softmax output layers, one for each training language. The authors show that for ASR, despite a lower word accuracy of multilingually trained features over baseline language-specific features, the multilingual features are more robust in conditions with non-native speakers.

The idea of language independent features based on universal speech attributes was investigated in Siniscalchi et al. (2013). They used manner and place of articulation to fully describe parts of speech in any language. In their SLR system, the sequences of these attributes were then modeled using a vector space modeling techniques. By using such articulatory features, there is no need for (language-dependent) phonetic transcriptions and the features can be also considered as language-independent.

The use of bottleneck features for SLR was investigated in Matějka et al. (2014); Jiang et al. (2014a). The authors of Matějka et al. (2014) report a 45% relative improvement to acoustic features baseline on DARPA RATS database. The authors of Jiang et al. (2014a) trained two deep bottleneck neural networks on English and Mandarin. The resulting features are then fused either on feature or score level.

Another approach to use neural networks in SLR was proposed by Lopez-Moreno et al. (2014, 2016): the NN was trained for frame-by-frame language classification. The final decision is based on language log-posteriors averaged over frames. This approach works great for short utterances. However, for long utterances the conventional i-vector approach (Lopez-Moreno et al., 2016) is still superior.

The use of Long Short Term Memory (LSTM) cells to directly classify languages has been investigated by Gonzalez-Dominguez et al. (2014); Zazo et al. (2016). The advantage of recurrent architectures is in natural handling of time context by memorizing internal state over time and also very small number of parameters compared to standard i-vector system. As it happens with other DNN approaches, this technique outperforms conventional i-vectors only in short durations.

A nice summary of neural network approaches for SLR can be found in Ferrer et al. (2016). The paper compares three types of features (SDC, bottleneck and probabilistic/posterior ones) that are modeled using two different approaches: standard GMM/UBM i-vector system and i-vector system using statistics collected using DNN alignment. The results show that for SLR, standard GMM/UBM i-vector system using bottleneck features performs the best.

1.2. Outline

The paper is organized as follows. We start with a description of bottleneck networks and bottleneck features. Multilingual training of NNs and its variants is introduced in Section 3. Experimental protocol, training and evaluation corpora and a description of our SLR system are in Section 4. The results are presented in Section 5, with conclusions in Section 6.

2. Bottleneck features

Bottleneck NN refers to such topology of a NN, where one of the hidden layers has significantly lower dimensionality than the surrounding layers Kramer (1991). It is assumed that such layer – referred to as the bottleneck – compresses the information needed for mapping the NN input to the output, making the system more robust to noise and over-fitting. A bottleneck feature vector is generally understood as a by-product of forwarding a primary input feature vector through the bottleneck network and reading the vector of values at the bottleneck layer. In other words, after a network is trained for its primary task (e.g., phone state classification), the bottleneck layer is declared to be the output layer and all succeeding layers are ignored. Such NN then maps the primary features to the bottleneck features.

Although we use the NN for SLR purpose, the NN itself is entirely trained for phone state classification. We do not use direct NN optimization for the SLR task, as is done, for example, in Lopez-Moreno et al. (2014). The whole SLR system is then run subsequently utilizing the trained NN just for bottleneck feature extraction.

We use context independent phoneme states as NN training targets for the output layer. The other possibility is to use more widely used senones. The comparison of these two unit types is given in Section 5.6. Both are taken from an alignment with a conventional HMM PLP-based LVCSR system.

2.1. Primary feature extraction

The NN input features are 24 log Mel-scale filter bank outputs augmented with fundamental frequency features. The fundamental frequency features consist of f_0 and probability of voicing estimates computed according to Talkin (1995), f_0 estimates obtained by Snack tool[1] function *getf0*, seven coefficients of Fundamental Frequency Variations spectrum according to Laskowski and Edlund (2010) and f_0 computed using Kaldi[2] with its delta coefficients and probability of voicing. Together, we have 13 f_0 related features, see Karafiát et al. (2014) for more details.

The conversation-side based mean subtraction is applied on the whole feature vector. 11 frames of log filter bank outputs and fundamental frequency features are stacked together. Hamming window followed by DCT consisting of 0th to 5th bases are applied on the time trajectory of each parameter resulting in $(24 + 13) \times 6 = 222$ coefficients on the NN input (see Fig. 1).

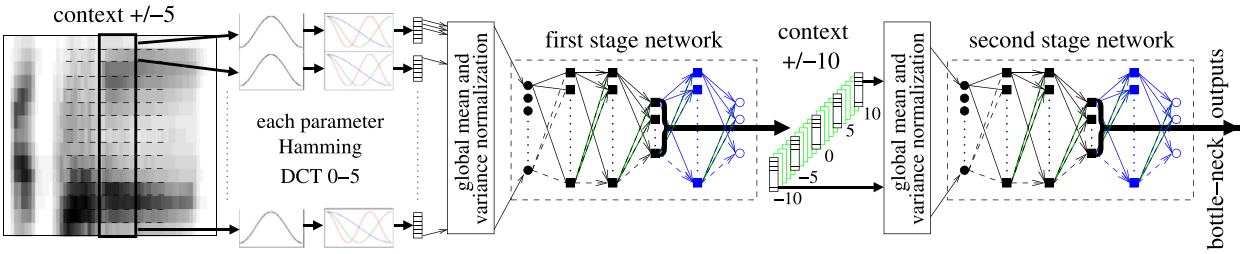


Fig. 1. Block diagram of Stacked Bottle-Neck (SBN) feature extraction. The blue parts of neural networks are used only during the training. The green frames in context gathering between the two stages are skipped. Only frames with shift –10, –5, 0, 5, 10 form the input to the second stage NN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

[1] http://www.speech.kth.se/snack/.
[2] http://kaldi.sourceforge.net.

2.2. Neural network architecture

The configuration for the feed-forward NN is $222 \times D_{HL} \times D_{HL} \times D_{BN} \times D_{HL} \times K$, where the K is the number of targets (see Table 6). The dimensionality of bottleneck layer, D_{BN}, was fixed to 80 in all experiments. This setting was shown as optimal in Matějka et al. (2014). According to our experience, larger dimensionality of bottleneck features (e.g., 80) tends to improve SLR system on short test segments and small bottleneck features give better performance for long ones. Therefore we use larger feature dimensionality to boost performance for very short segments. This phenomenon is clearly a consequence of available feature resolution. With conservative dimensionality, the information about phonetic content is predominant in feature space. By increasing the size of the bottleneck layer, we allow the other variability present in signal (like speech prosody or noise information) to be more widely represented in features. This can help to distinguish between very short segments where sufficient phonetic content is scarce.

The dimensionality of full hidden layers, D_{HL}, was set to 1500, if not stated otherwise. We use logistic sigmoids as non-linearities for these layers. The bottleneck layer has a linear activation function which was shown to provide good performance (Veselý et al., 2011). As discussed in Kramer (1991), bottleneck layer can be linear without loss of generalization, because the non-linear mapping transformations are already covered by full hidden layers.

2.3. Stacked Bottleneck features

In most of the experiments, we have used a cascade of two bottleneck NNs (see Fig. 1). The outputs of the first network are *stacked* in time, defining broader context input features for the second NN, hence the term stacked bottleneck features (SBN).

This stacked architecture was designed specifically for the ASR task (Grézl et al., 2009, 2013). The first BN network operates on a short context of several frames (typically 11 frames, see Section 2.1), producing input features for the second network. These are sampled at times $t-10$, $t-5$, t, $t+5$ and $t+10$, where t is the index of the current frame. As there are 11 acoustic frames on the input of the first stage NN, this 1:5 sub-sampling corresponds to one-half context (5 frames) overlap and results in total context of $4 \times 5 + 11 = 31$ frames (325 ms). The resulting $5 \times 80 = 400$-dimensional features are input to the second stage NN with similar configuration of $400 \times D_{HL} \times D_{HL} \times D_{BN} \times D_{HL} \times K$.

The 80 bottleneck outputs from the first NN (referred to as *BN* features) or from the second NN (referred to as *SBN* features) are taken as features for the conventional GMM UBM i-vector based SLR system described below in Section 4.3.

3. Multilingual training of neural networks

The basic idea is to train NN in a multilingual manner – on more than one language (Schultz and Waibel, 2001), so that the final bottleneck features cover richer acoustic space than when trained for one language only. There are several possible multilingual training schemes based on a setup of the NN output layer:

- *One softmax* – the output layer of NN is created as a concatenation of language-tagged[3] phoneme sets for individual languages, without any mapping or clustering of individual phonemes. It discriminates between all target units of all languages. The resulting NN has fairly large output layer containing all speech units from all languages with a single softmax. From our previous experiments in ASR (Veselý et al., 2012), we know that the single softmax approach does not work well, as very similar phones from different languages are considered as different targets and part of the "bottleneck encoding capacity" is wasted to discriminate between them. This evidently holds for the ASR. For the SLR, however, the single softmax scheme could be a better option, as the network has to learn (internally), how to discriminate between languages and choose the target unit from an appropriate language.
- *IPA mapping* – the per-language phoneme sets are mapped to a global phoneme set according to prior expert phonetician knowledge or by means of clustering using a confusion matrix. Similar approach to what we describe in

[3] Tagged for example like English_A, German_A, Turkish_A ...

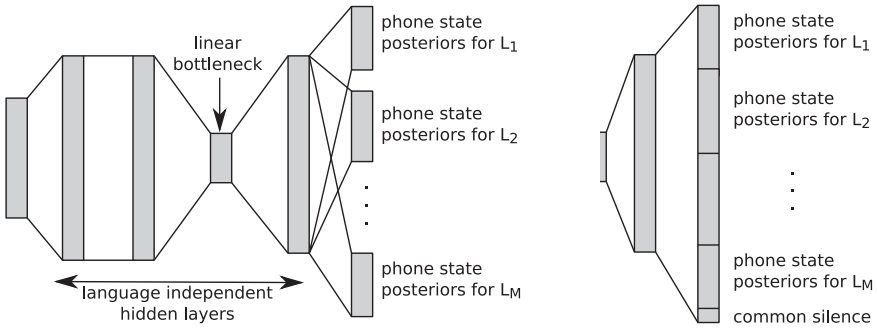


Fig. 2. Schematic illustration of block and one softmax output layer for multilingual training.

this paper is presented in Dupont et al. (2005). We do not consider this scheme in our work. See Schultz (2002) for more detailed description.

- *Block softmax* – divides the output layer into parts according to individual languages (Veselý et al., 2012). During the training, only the part of the output layer corresponding to the language of the actual target is activated. We use this block softmax approach in our experiments because of the better performance (see Section 5.4).

  As there is an inductive transfer between output blocks and data representation is shared among languages, this approach is very similar to multi-task learning (MTL) (Caruana, 1997). With only a small difference: in MTL, there are output targets of all blocks available for each training sample, whereas in block softmax approach there is only one block active per training sample.

Simplified illustrations for *one* and *block* softmax output layers are in Fig. 2. Detailed description can be found in Veselý et al. (2012).

For the sake of completeness, there are other possibilities to obtain a multilingual system. It is always possible to build several parallel systems trained on specific languages and then fuse them on feature or score level. This was used, for example, in successful P-PRLM approach (Zissman and Singer, 1994), where multiple streams from different phoneme recognizers were used to form N-gram statistics fused in score domain.

A disadvantage of such approaches is the computational cost, as many systems must run in parallel. In our case, the block-softmax approach performs such a parallelization on the level of the output layer, and rather than hand-merging the phone inventories (IPA mapping) it manages to fuse the phone systems of individual languages by the internal representation of the network.

4. Experimental setup

4.1. Multilingual training data

For training the neural networks, we used mainly the IARPA Babel Program data.[4] This data simulates a case of what one could collect in limited time from a completely new language. It consists mainly of telephone conversational speech (CTS), but scripted recordings, as well as far field recordings, are present as well.

All training data from the full language packs from the collections shown in Table 1 were used in our experiments. More details about the characteristics of the individual languages can be found in Harper (2013). The speech was force-aligned using our Babel ASR system (Karafiát et al., 2013).

For NN training, we used either one language (monolingual networks) or subsets of Babel data with different number of languages. We denote these multilingual training sets according to number of languages used as *multi5, multi6, multi11* and *multi17*. See Table 1.

[4] Collected by Appen, http://www.appenbutlerhill.com.

Table 1
Data used to train bottleneck networks. The proportion of silence varies from 15 to 20% depending on language.

Source	Language				# hours
Babel	Cantonese	**multi5** (284h)	**multi11** (641h)	**multi17** (1070h)	65.0
Babel	Pashto				64.7
Babel	Turkish				56.6
Babel	Tagalog				44.1
Babel	Vietnamese				53.2
Babel	Assamese	**multi6** (357h)			46.7
Babel	Bengali				53.6
Babel	Haitian Creole				55.0
Babel	Lao				71.6
Babel	Tamil				72.7
Babel	Zulu				57.8
Babel	Kurdish				69.7
Babel	Tok Pisin				68.7
Babel	Cebuano				70.8
Babel	Kazach				73.0
Babel	Telugu				71.7
Babel	Lithuanian				81.4
Fisher	English 60h				60
Fisher	English 250h				250
Fisher	English 2043h (full Fisher)				2043

For experiments where we needed a lot of training data for a single language, we used Fisher English Training Part 1/2, containing 2043 h of clean speech. We created two random subsets of lengths 60 h and 250 h to simulate the average length of language packs from Babel and the length of the multi5 set, respectively.

4.2. NIST LRE 2009 evaluation database

We present our results on NIST LRE 2009 database in terms of average detection cost as defined in NIST (2009). To train our systems (parameter estimation for UBM, total variability subspace and backend), we used the same data setup as in Jančík et al. (2010): Callfriend, Fisher English Part 1 and 2, Fisher Levantine Arabic, HKUST Mandarin, Mixer (data from NIST SRE 2004, 2005, 2006, 2008), Foreign Accented English, OGI Multi-Language Telephone Corpus, OGI 22 languages, Voice of America radio broadcasts and development data for LRE 2005 and LRE 2007.

Four datasets were used for training:

- *full54* – all data from the databases (54 languages, 75 thousand files, 2500 h of speech). This was used for i-vector subspace training.
- *full23* – subset of 23 target languages for LRE09 from the full54 set (23 languages, 51 thousand files, 1550 h). This was used to train backend classifier.
- *balanced23* – maximum of 500 utterances were selected from full23 set for each language (23 languages, 9.8 thousand files, 360 h). This balanced set was used for UBM training.
- *devel* – data from all pre-2009 NIST LRE evaluations, OGI-multilingual, OGI 22 languages, Foreign Accented English, SpeechDat-East, Switchboard and VoA broadcasts (23 languages, 38.5 thousand files, 117 h). This set was used for calibration/fusion parameter estimation.

4.3. SLR system description

We based our experiments on the i-vector based system (Martínez et al., 2011). I-vectors provide an elegant way of reducing the variable-length input data (time sequence of features) to a small fixed-dimensional feature vector while retaining most of the relevant information (Dehak et al., 2010). Combination of the i-vector paradigm with bottleneck features is a state-of-the-art technique for SLR (Ferrer et al., 2016; Jiang et al., 2014b; Richardson et al., 2015).

4.3.1. Feature extraction

For the bottleneck feature extraction, please refer to Sections 2 and 3, where it is described in detail. As the reference features, we used SDC features (Torres-Carrasquillo et al., 2002) with usual configuration 7-1-3-7, concatenated with 7 original MFCC coefficients (including C0). The frame rate is 10 ms. Vocal Tract Length Normalization (VTLN) (Welling et al., 1999), Cepstral mean and variance normalization (CMVN) and RASTA filtering (Hermansky and Morgan, 1994) was applied before SDC.

After feature extraction, voice activity detection (VAD) was performed by our Hungarian phoneme recognizer – we simply drop all frames that are labeled as silence or speaker noises.

4.3.2. Estimation of total variability subspace

Our i-vector extractor was trained in 5 iterations of jointly applying the Expectation-Maximization (EM) algorithm and the Minimum Divergence (MD) step (Brummer, 2014). If not stated otherwise, sufficient statistics for i-vector extraction were collected using a 512 component GMM with diagonal covariance matrices trained on *balanced23* set. The i-vector dimensionality was set to 400.

4.3.3. SLR backend

We used Gaussian classifier with shared covariance matrix (Martínez et al., 2011) trained on our *full23* set to produce language log-likelihoods. The shared covariance matrix was computed as an average of covariance estimates for each class. The motivation for this was a class imbalance in the training set. It is in principle the same as a weighted Gaussian backend described in Ferrer et al. (2016). This classifier was also used to calibrate the scores of individual systems on the development set.

The fusion was performed by training a linear logistic regression, where every subsystem has a tunable scalar weight and every target language a tunable scalar offset. The fusion can be written as:

$$\boldsymbol{s}_k = \sum_{m=1}^{M} \alpha_m \boldsymbol{s}_{km} + \boldsymbol{\beta},$$

where M is the number of subsystems to fuse, $\boldsymbol{s}_{km}$ is vector of calibrated log-likelihoods for segment k and subsystem m and α and β are fusion parameters (scale and offset). These parameters are trained on the development set to minimize the multiclass cross-entropy (Van Leeuwen and Brummer, 2006). The trial weights (appearing in cross-entropy objective function) are set again to be inversely proportional to class frequencies, to compensate for language imbalance.

It is important to optimize the fusion parameters α and β also for single systems (i.e., train one-way fusion). Otherwise, the comparison of single and fused systems is biased towards fused ones because of missing final discriminative score calibration (training a fusion for one single system can be regarded as a score calibration).

5. Results

5.1. Multilingual features

We performed a set of experiments to see how multilingual training affects the SLR performance of SBN features. Intuitively, we should see an improvement, as more different speech units are used as targets during NN

Table 2
Comparison of systems based on monolingual and multilingual SBN features. All the NNs were trained using the same network architecture (except for the difference in the output layer). We used 2048 component diagonal UBM for the experiments in this table.

ID	Features	$C_{avg} \times 100$		
		3 s	10 s	30 s
sdc	SDC	14.68	4.66	2.16
1	Cantonese	10.77	3.12	1.54
2	Tagalog	10.45	3.03	1.51
3	Vietnamese	10.27	3.00	1.48
4	Turkish	9.53	2.71	1.42
5	Pashto	9.17	2.54	1.33
	English 60 h	8.92	2.43	1.20
	English 250 h	8.74	2.35	1.15
	English 2043 h	8.43	2.20	1.15
m5	multi5 (284 h)	7.17	2.01	1.14
m6	multi6 (360 h)	6.81	1.83	1.08
m11	multi11 (640 h)	6.71	**1.80**	**1.05**
m17	multi17 (1070 h)	**6.44**	1.81	**1.05**

training, which should lead to the ability to better discriminate languages. Moreover, it was already proven to be effective for ASR (Veselý et al., 2012) so the expectation was high.

We trained five monolingual systems for each language from the multi5 set and one multilingual system using all data from this set, see Table 2.

Note, that all monolingual systems for languages from multi5 set use less data (cca 60 h each) compared to multilingual system multi5 (284 h) and multi11 (640 h). We did not have more data for these languages, so to do a fair comparison, we trained another monolingual network on a comparable random subset of Fisher English database. We can see from Table 2, that the multi5 system outperforms the monolingual system (Fisher 250 h) by more than 10% relative. We also show a system trained on a 60 h subset of Fisher English, that is, in turn, comparable to other monolingual systems from the multi5 set. This system is superior to all monolingual systems from the multi5 set. Finally, we show results for the system trained on all data from Fisher database (2043 h). Especially for short test durations, we can see, that the multilingual features are still superior.

5.2. Analysis of monolingual features

Based on results from previous section, it would be hard to select the best single language for training the monolingual NN. Some of the training languages are target languages for LRE09 and proportions of test trials for different target languages vary a lot. For example, English SBN features trained on the 60h subset of Fisher seem to be the best choice among monolingual features, but this is biased by the fact that English is dominant target language for LRE09 and probably because of a better quality of transcriptions in Fisher database. In this section we analyze the behavior of monolingual features further. Firstly we show how the language used to train bottleneck features affects per-language SLR scores. Secondly we show how the language used affects the variance of resulting i-vectors. We do this to illustrate the weaknesses of monolingual features and to motivate the multilingual training paradigm.

5.2.1. Per-language SLR performance of bottleneck features

The main goal of multilingual training is to produce language-independent or "universal" features. In Fig. 3 we show how per-language recognition performance behaves for language-dependent (monolingual) and multilingual features. We can see that for these 5 target languages, the best performance is achieved by using SBN features trained on the same language. This makes sense and demonstrates the language bias of monolingual features.

If we compare this best performance with that of multilingual features, we can see that they can be very similar. For example, the best SLR performance on Pashto language using monolingual features is achieved with Pashto

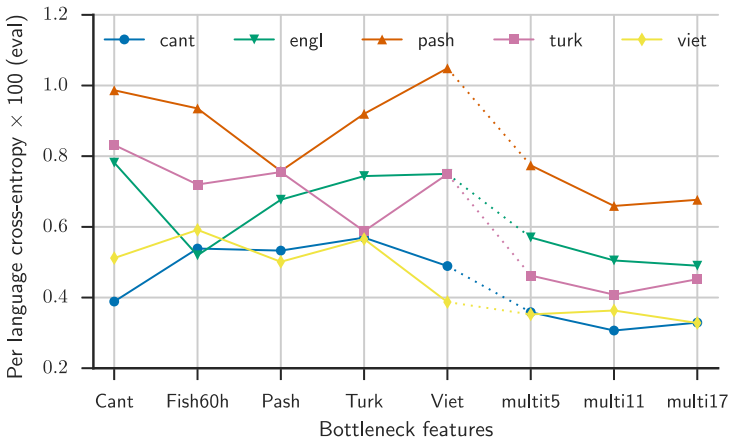


Fig. 3. Per-language multiclass cross-entropy (lower is better) evaluated for different SBN features. We show only the languages from LRE09 matching with languages used to train monolingual SBN features. Dotted lines are here to separate mono- and multi-lingual features.

features. And that this is almost equal to the performance of multi5 features on that language. The line reaches minimum (the best performance) for multi11 SBN features.

For some languages we can see a degradation of performance when using more languages in multilingual training of SBN features (see per-language scores for multi11 and multi17 features in Fig. 3). This is probably due to a lower proportion of this particular language in multilingual training dataset. Regardless of this, overall performance gets better with more languages added to training (see Table 2).

5.2.2. *Differences in variance of i-vectors*

Interestingly, the variance of i-vectors for a language that was used for NN training can increase. The analysis of this behavior is illustrated in Fig. 4. We used i-vectors from the development set, as it is balanced in terms of duration and number of segments. The i-vectors for individual features were normalized to have zero mean and unit variance (mean and variance was computed on all data, i.e., on merged train, devel and eval set). To measure differences in within-class i-vector distribution, we used average variance of i-vectors of given language, computed as:

$$\sigma^{l}_{avg} = \frac{trace(\Sigma^{l}_{wc})}{D},$$

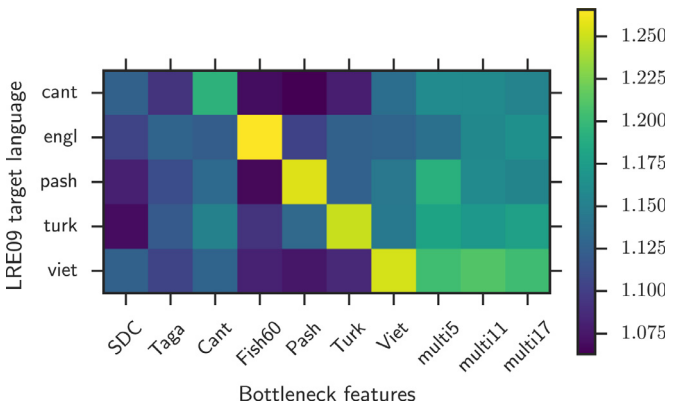


Fig. 4. Average variance of i-vectors for a subset of languages from LRE09 devel set (vertical axis) generated from SBN features trained on different languages (either mono- or multi- lingually). The SDC features are also shown in the first column. Note the diagonal pattern when languages match.

where Σ_{wc}^{l} is within-class covariance matrix for language l and D (i-vector dimensionality, in our case 400) is used to normalize the sum inside the trace.

We can then compare these variances among languages and features. Fig. 4 displays only selected subset of LRE09 target languages, that were also available to us for SBN NN training (Cantonese, Fisher English, Pashto, Turkish, Vietnamese). We can clearly see a diagonal pattern, which tells us, that i-vectors are more variable for languages matching the ones used to train SBN feature extractor.

For the multilingually trained SBN features, we can see that the variance is more uniform across target languages.

The first two columns in Fig. 4 display also i-vector variances for SDC features (which should be a good reference for language-independent features) and SBN features trained on the Tagalog language from Babel. Tagalog is not a target language in LRE09 and we can see the expected behavior: the i-vectors extracted from these features are more or less "uniform" across our 5 language subset.

We assume that increased i-vector variance is caused by richer feature space for linguistically matching phrases. This then propagates to i-vector space through the UBM alignment statistics. In practice, we want to avoid differences in i-vector variance for different languages as this violates our assumption about shared within class covariance of i-vectors in case of linear Gaussian backend.

5.3. Fusion of mono- and multi-lingual systems

By looking at the gains from adding different systems to the fusion, we can roughly estimate the complementary information conveyed by different features. Note that the gain is not solely dependent on features. Each subsystem has different UBM alignment and different i-vector subspace (both UBM and i-vector subspace training start from random initialization and are not guaranteed to converge to global minimum), which necessarily extracts different and possibly complementary cues about the recording.

Table 3 shows the fusion of five monolingual systems (F1) and the same fusion with multi5 system included (F2). When we compare the multi5 multilingual system and the fusion of five corresponding monolingual systems (first two lines in Table 3), we can see that the five-fold fusion performs better across all conditions. Having several specialized (language dependent) features seems to be more appropriate than using some universal (language independent) feature set. On the other hand, the single multi5 system is 5 × smaller. Interestingly, we can see from fusion F2, that the multi5 system brings on average additional gain of 5% to the fusion F1, which demonstrates that multilingual features convey yet another kind of information to the final fusion.

We also tried adding the MFCC/SDC system to the fusion to find out, whether standard cepstral features still contain useful complementary information. As we can see from fusions F3, F4a and F4b, this yields only minor improvement, mainly for short durations. Such behavior can be expected, as a system based on spectral features may fetch valuable information when the duration of test segment is limited to just a few seconds.

5.4. One softmax

Multilingual training of NN using block softmax (as described in Section 3) was originally designed for ASR. The network is trained to predict posterior probabilities of speech units (phoneme states in our case). By using the block softmax output layer, we are able to train NN on different phoneme sets. Each of these phoneme sets contains unique

Table 3
Selected fusions of systems based on monolingual and multilingual SBN features from Table 2.

	System combination	$C_{avg} \times 100$		
		3 s	10 s	30 s
	m5 /single system/	7.17	2.01	1.14
F1	1+2+3+4+5	6.38	1.87	1.12
F2	1+2+3+4+5 + m5	6.10	1.73	**1.07**
F3	1+2+3+4+5 + m5 + sdc	**5.92**	**1.66**	**1.07**
F4a	1+2+3+4+5 + sdc	6.34	1.83	1.13
F4b	m5 + sdc	6.61	1.85	1.10

Table 4
Comparison of block- and one-softmax output layers for multilingual training on multi5 set. We used 30-dimensional bottleneck layer for the stacked architecture (SBN30). For the single stage experiments (BN80), we use 80-dimensional features with input temporal context increased to 51 frames.

Features	Output	$C_{avg} \times 100$			
		3 s	10 s	30 s	All
SBN30, 31 frames	Block	9.62	2.54	1.30	4.46
SBN30, 31 frames	One	10.45	2.85	1.42	4.88
BN80, 51 frames	Block	9.18	2.57	1.37	4.35
BN80, 51 frames	One	10.18	2.86	1.54	4.84

units, but they are not necessarily disjoint between the sets. If we used just one softmax with targets concatenated from phoneme sets of multiple languages (see Fig. 2b), then the network would need to extract also some language discriminating information from the temporal context and encode it into bottleneck activations to be able to properly select the correct output unit. This information, encoded in bottleneck features, could be advantageously used in SLR. The motivation is similar to that of anchor models (Sturim et al., 2001).

Earlier experiments with this type of output in ASR domain have shown its inferiority to block softmax approach (Veselý et al., 2012). Our results in Table 4 indicate, that this holds also when the bottleneck features are used for SLR. It is clear that during the training, the NN is not able to handle situations when two similar units from different languages compete. This is even worse when one softmax is used in the stacked architecture.

For these reasons we tried to limit the network to the first stage only (no stacking) and we also tried to increase the input context (from 31 to 51 frames). The results are in the second part of Table 4. We see that even with this modified network, the one-softmax output does not give us any improvement over the block-softmax. Both NN configurations (SBN with the default context and BN with increased context) suffer about the same relative degradation from using the one-softmax output layer.

5.5. *Input context for BN and SBN features*

As we have already mentioned, the hierarchical two-stage topology of Stacked Bottleneck (SBN) NN was originally designed for ASR task, where it outperformed the single-stage configuration (Grézl et al., 2009). We performed a set of experiments to see SLR performance of bottleneck features from both single-stage (BN) and two-stage (SBN) networks. Different configurations can be directly compared by looking at the total temporal context the networks have access to.

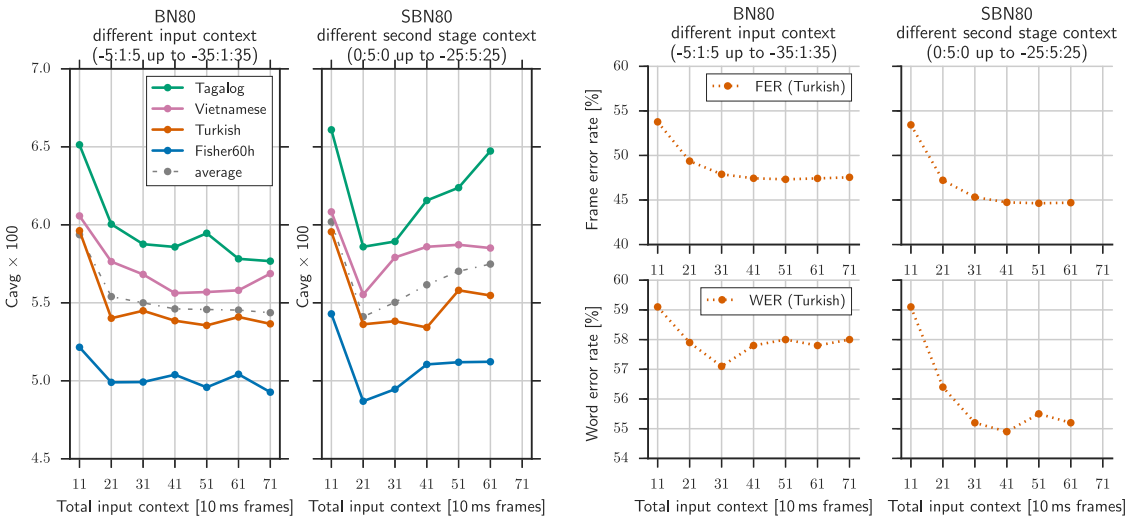
We shall start with a description of our notation we use to denote input temporal context for NN. Three numbers separated by colon denote the number of frames for left context, step and number of frames for right context. For example, –5:1:5 means that together with central frame, we take 5 frames of the left context and 5 frames of the right context and we take every frame (no sub-sampling), resulting in 11 frames.

The second stage network works effectively as a merger of the first stage network bottleneck features from different regions of time context. As the first network sees 125 ms of time context with 10 ms frame shift, we sub-sample these first stage bottleneck features 1:5 and stack them again to extend the global context to about 300 ms. The information would be highly redundant without the sub-sampling.

The same context span can be accessed with only one NN simply by correspondingly extending its input. This results in a smaller network with input temporal context span preserved. On the other hand, it leads to more shallow network, without any hierarchy and possibility of parameter sharing. The results are shown in Fig. 5. For the two-stage SBN features, the position on the x-axis was determined as the largest time context the network has access to (overall context, measured in frames).

We can see that the difference between stacked architecture and simpler single stage NN is not big in terms of SLR performance. In several cases, the SBN features outperform the BN features, but on average simpler BN NN

[5] Note that the frame error rates were taken from networks in the final training epoch. The plot thus does not display training evolution on cross-validation set for different epochs (as usually), but in our case for networks with different input temporal context.



(a) SLR performance

(b) ASR-related metrics

Fig. 5. Performance of BN and SBN features with different temporal context. On the left the figure shows SLR performance of 4 monolingual systems (BN NNs trained on Tagalog, Vietnamese, Turkish and 60h subset of Fisher English). The right part shows ASR-related metrics: final frame- and word-error rates on cross-validation set (only for the Turkish language).

with increased context performs almost the same. On the other hand, both ASR metrics (right part of Fig. 5, only Turkish features are shown) are apparently better for the stacked NN architecture.

For the single stage BN features, there is a significant improvement in C_{avg} between –5:1:5 (11 frames) and –10:1:10 (21 frames) and except Vietnamese, BN features seem to get progressively better when using longer temporal context. This is not the case for the stacked architecture, as the performance of SBN features starts getting worse when using more than 21 frames of (stacked) global context.

Interestingly, when we look at the final frame error rates (FER) of the Turkish NNs (top-right part of Fig. 5), we can see that adding more frames of temporal context to the NN input helps to improve the per-frame accuracy[5]. The FER and WER on cross-validation set seems to saturate when using input context of 31 frames and more (that is why 31 frames is our default context for the SBN architecture).

5.6. *Using senones as NN training targets*

Our experiments in Matějka et al. (2014) and Fér et al. (2015) have shown, that we can obtain the best features by using senones (context-dependent tied triphone states) as NN training targets. Senones also dominate other recent works making use of bottleneck NNs (Jiang et al., 2014a; Ferrer et al., 2016). In the case of senones, however, the output layer can become very large when training the NN multilingually on many languages. It can even exceed the size of the whole rest of the network several times (see Fig. 6), requiring a lot of memory during the NN training. This is because the number of senones is an order of magnitude higher compared to the number of phoneme states (see Table 6).

In this section, we show that the amount of parameters in the output layer can be lowered down by moving the bottleneck layer to the very end of a network, so the block-softmax output is immediately following the bottleneck. This eliminates a majority of the weights in the output layer, but also limits the decoding power of a bottleneck-to-softmax transformation to be just linear . We show that this can degrade the SLR performance, mainly for short test segment durations.

Note that placing the bottleneck to the end of a network was shown to be disadvantageous in McLaren et al. (2016). The authors evaluate five-layer NN with the bottleneck layer at different positions, both on LRE09 and noisy RATS database. They achieved the best results on LRE09 with bottleneck layer being the last but one in the network. For the noisy RATS data, positions in the center were found better.

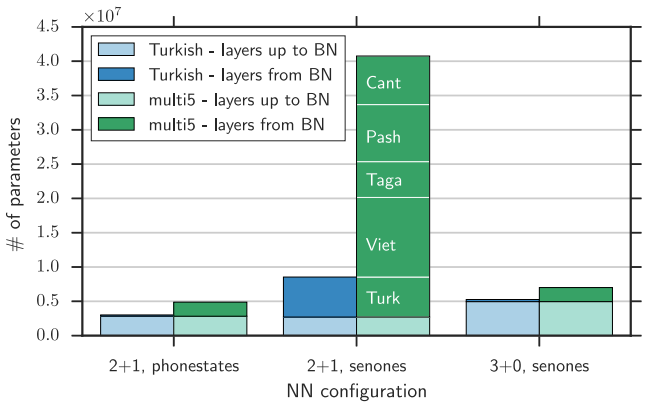


Fig. 6. Amounts of free parameters of BN NN for different organizations of hidden layers (only the first stage of the SBN architecture is considered). Note that the light parts of both green and blue bars do not change for different output layer (either mono- or multi-lingual). The darker parts of bars represent units of bottleneck NN not used during the bottleneck feature extraction. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Considering phoneme states as our baseline NN training targets, we evaluate two systems based on senones. One with a full hidden layer between bottleneck and output layer (denoted as "2+1" – the same topology as in baseline) and a second system with proposed modification where the bottleneck layer is shifted to the very end of a network (denoted as "3+0"). The results are in Table 5.

We can see that in the case of monolingual Turkish features, phoneme states as targets do not perform as good as senones and that shifting bottleneck layer to the end of the network seems to have a minor effect on SLR performance, with a degradation for short segments only. In the case of multilingual features, the situation is similar.

Table 5
Experiments with different training targets for monolingual and multilingual training (SBN architecture). The number of hidden layers *before* and *after* the bottleneck is indicated in the second column.

Targets	Full layers	$C_{avg} \times 100$			
		3 s	10 s	30 s	All
Turkish phonestates	2+1	9.53	2.71	1.42	4.53
Turkish senones	2+1	**8.72**	2.40	1.27	**4.11**
Turkish senones	3+0	9.02	**2.39**	**1.26**	4.20
multi5 phonestates	2+1	7.17	2.01	1.14	3.42
multi5 senones	2+1	**7.09**	**1.90**	**1.06**	**3.33**
multi5 senones	3+0	7.15	1.94	1.07	3.37

Table 6
Number of output layer targets for individual training languages and different kinds of speech units.

	# Phones	# Phonestates	# Senones
Turkish	42	126	3805
Pashto	72	216	5541
Tagalog	84	252	3475
Vietnamese	101	303	7731
Cantonese	157	471	4718
multi5	456	1368	25,270

Comparing the two network topologies based on senones, the system with layer configuration "2+1" is the best over all conditions. The second configuration ("3+0"), especially for short segments, yields performance closer to the system based on phoneme states. The use of senones as targets in multilingually trained networks thus still leads to very large networks if we do not want to sacrifice any performance on short duration test segments. This might not be a problem if we consider that the biggest part of a network can be discarded once the NN is trained for bottleneck feature extraction.

Another approach to scale down the number of parameters in the last layer could be forcing the state-tying algorithm to cluster the triphone states to a very small number, comparable to the number of phonestates. Note that it would probably end up with a very similar set of targets, most probably falling back to the same performance as with phonestates.

6. Conclusions

In this work, we applied multilingual training paradigm of neural networks to extract rich features for SLR. On the standard NIST LRE09 dataset, we have shown that features enriched in this way are more informative and better fitting for a SLR task than monolingual ones. This corresponds with original results from ASR domain.

We showed that the multilingual features are complementary to the fusion of corresponding monolingual systems as they bring additional gain in fusion. This confirms the hypothesis, that we can obtain more complex information from multilingual bottleneck features than from fusion of "simple" parallel systems.

We briefly investigated the suitability of training the NN using only concatenated phoneme sets (one softmax output). Although the final results are worse (compared to block softmax output), they are surprisingly good and comparable to that of monolingual features. Our initial hypothesis that a network would be able to discriminate between similar targets of different languages (based on such a short temporal context) turned out to be wrong. This confirms a concept of poly-phonemes (phonemes common to more languages) as described in Anderson et al. (1994).

A large part of the article was dedicated to practical aspects of multilingual training of bottleneck networks. We showed and analyzed the main pitfalls arising from the increased size of the output layer for senones as targets and we focused on the choice of the size of the input temporal context. The results were given in conjunction with per-frame NN training objective to put more insight into the relation of SLR performance and NN training accuracy. The results indicate, that even thought in ASR domain the stacked SBN architecture (used throughout this paper) performs better than single stage architecture, SLR performance comparable to the stacked SBN architecture can be achieved using only the first stage bottleneck network with extended input temporal context.

Per-language results showed strong coupling of monolingual features to their training language, i.e., that language-specific features produce the best scores for that particular language. Multilingual features performed either that well or better, consistently and independently of the language. Note that ASR experiments in Scanzio et al. (2008) resulted in the opposite behavior. In that paper, a baseline monolingual features trained on a language matching a test language performed better than a multilingually trained features. These "specialized" features were simply better fitting to that particular language. This does not apply for SLR, where the task is inherently multilingual. Our subsequent analysis of i-vector variance demonstrated the language-independence of multilingually trained bottleneck features and suggested that such features are more appropriate when using a linear Gaussian backend.

Acknowledgments

This work was supported by the Czech Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project "IT4Innovations excellence in science – LQ1602".

References

Anderson, O., Dalsgaard, P., Barry, W., 1994. On the use of data-driven clustering technique for identification of poly- and mono-phonemes for four European languages. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, vol. I, pp. I/121–I/124.

Brummer, N., 2014. The EM Algorithm and Minimum Divergence. Technical Report. Agnitio Labs.

Caruana, R., 1997. Multitask learning. Mach. Learn. 28 (1), 41–75.

Corredor-Ardoy, C., Gauvain, J., Adda-Decker, M., Lamel, L., 1997. Language identification with language-independent acoustic models. In: Proceedings of the Eurospeech, pp. 355–358.

Dehak, N., Kenny, P., Dehak, R., Dumouchel, P., Ouellet, P., 2011. Front end factor analysis for speaker verification. IEEE Trans. Audio Speech Lang. Process 19 (4), 788–798.

Dupont, S., Ris, C., Deroo, O., Poitoux, S., 2005. Feature extraction and acoustic modeling: an approach for improved generalization across languages and accents. In: Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding, pp. 29–34. doi: 10.1109/ASRU.2005.1566527.

Fér, R., Matějka, P., Grézl, F., Plchot, O., Černocký, J., 2015. Multilingual bottleneck features for language recognition. In: Proceedings of the Interspeech. Dresden, Germany, pp. 389–393.

Ferrer, L., Lei, Y., McLaren, M., Scheffer, N., 2016. Study of senone-based deep neural network approaches for spoken language recognition. IEEE/ACM Trans. Audio Speech Lang. Process. 24 (1), 105–116.

Gonzalez-Dominguez, J., Lopez-Moreno, I., Gonzalez-Rodriguez, J., Moreno, P.J., 2014. Automatic language identification using long short-term memory recurrent neural networks. In: Proceedings of the Interspeech. Singapore.

Grézl, F., Egorova, E., Karafiát, M., 2014. Further investigation into multilingual training and adaptation of stacked bottle-neck neural network structure. In: Proceedings of the Spoken Language Technology Workshop (SLT). South Lake Tahoe, Nevada USA.

Grézl, F., Karafiát, M., Burget, L., 2009. Investigation into bottleneck features for meeting speech recognition. In: Proceedings of the Interspeech. Brighton, England.

Grézl, F., Karafiát, M., Kontár, S., Černocký, J., 2007. Probabilistic and bottle-neck features for lVCSR of meetings. In: Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 4, pp. IV-757–IV-760.

Grézl, F., Karafiát, M., Veselý, K., 2013. Adaptation of neural network feature extractor for new language. In: Proceedings of the Interspeech. Lyon, France.

Harper, M., 2013. The BABEL program and low resource speech technology. In: Proceedings of the Automatic Speech Recognition and Understanding.

Hermansky, H., Morgan, N., 1994. Rasta processing of speech. IEEE Trans. Speech Audio Process. 2 (4), 578–589.

Jančík, Z., Plchot, O., Brummer, N., Burget, L., Glembek, O., Hubeika, V., Karafiát, M., Matějka, P., Mikolov, T., Strasheim, A., Černocký, J., 2010. Data selection and calibration issues in automatic language recognition – investigation with BUT-AGNITIO NIST LRE 2009 system. In: Proceedings of the Odyssey 2010 – The Speaker and Language Recognition Workshop, pp. 215–221.

Jiang B, Song Y, Wei S, Liu J-H, McLoughlin IV, Dai L-R (2014) Deep Bottleneck Features for Spoken Language Identification. PLoS ONE 9(7): e100795. https://doi.org/10.1371/journal.pone.0100795.

Jiang, B., Song, Y., Wei, S., Wang, M.G., McLoughlin, I., Dai, L.R., 2014. Performance evaluation of deep bottleneck features for spoken language identification. In: Proceedings of the Ninth International Symposium on Chinese Spoken Language Processing.

Karafiát, M., Grézl, F., Hannemann, M., Veselý, K., Černocký, J.H., 2013. BUT BABEL System for Spontaneous Cantonese. In: Proceedings of the Interspeech. Lyon, France, pp. 2589–2593.

Karafiát, M., Grézl, F., Veselý, K., Hannemann, M., Szőke, I., Černocký, J., 2014. BUT 2014 Babel system: analysis of adaptation in NN based systems. In: Proceedings of the Interspeech, pp. 3002–3006.

Kramer, M.A., 1991. Nonlinear principal component analysis using autoassociative neural networks. AIChE J. 37 (2), 233–243. doi: 10.1002/aic.690370209.

Laskowski, K., Edlund, J., 2010. A Snack implementation and Tcl/Tk interface to the fundamental frequency variation spectrum algorithm. In: Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). Valletta, Malta.

Lopez-Moreno, I., Gonzalez-Dominguez, J., Martinez, D., Plchot, O., Gonzalez-Rodriguez, J., Moreno, P.J., 2016. On the use of deep feedforward neural networks for automatic language identification. Comput. Speech Lang. 40, 46–59.

Lopez-Moreno, I., Gonzalez-Dominguez, J., Plchot, O., 2014. Automatic language identification using deep neural networks. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing. Florence, Italy.

Martínez, D.G., Plchot, O., Burget, L., Glembek, O., Matějka, P., 2011. Language recognition in ivectors space. In: Proceedings of the Interspeech, pp. 861–864.

Matějka, P., et al., 2014. Neural network bottleneck features for language identification. In: Proceedings of the IEEE Odyssey: The Speaker and Language Recognition Workshop. Joensu, Finland.

McLaren, M., Ferrer, L., Lawson, A., 2016. Exploring the role of phonetic bottleneck features for speaker and language recognition. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing. Shanghai, China.

NIST, 2009. The 2009 NIST Language Recognition Evaluation Plan (LRE09). Technical Report. National Institute of Standards and Technology.

Richardson, F., Reynolds, D., Dehak, N., 2015. A unified deep neural network for speaker and language recognition. In: Proceedings of the Interspeech. Dresden, Germany.

Scanzio, S., Laface, P., Fissore, L., Gemello, R., Mana, F., 2008. On the use of a multilingual neural network front-end. In: Proceedings of the Interspeech. Brisbane, Australia, pp. 2711–2714.

Schultz, T., 2002. GlobalPhone: a multilingual speech and text database developed at Karlsruhe University. In: Proceedings of the International Conference on Spoken Language Processing, pp. 345–348.

Schultz, T., Waibel, A., 2001. Language independent and language adaptive acoustic modeling for speech recognition. Speech Commun. 35, 31–51.

Siniscalchi, S.M., Reed, J., Svendsen, T., Lee, C.-H., 2013. Universal attribute characterization of spoken languages for automatic spoken language recognition. Comput. Speech Lang. 27 (1), 209–227. Special issue on Paralinguistics in Naturalistic Speech and Language, http://dx.doi.org/10.1016/j.csl.2012.05.001.

Sturim, D., Reynolds, D., Singer, E., Campbell, J., 2001. Speaker indexing in large audio databases using anchor models. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing. IEEE, pp. 429–432.

Talkin, D., 1995. A robust algorithm for pitch tracking (RAPT). In: Proceedings of the Speech Coding and Synthesis. Elsevier, New York.

Torres-Carrasquillo, P., Singer, E., Kohler, M., Greene, R., Reynolds, D., Deller, J., 2002. Approaches to language identification using gaussian mixture models and shifted delta Cepstral features. In: Proceedings of the International Conference on Spoken Language Processing, pp. 89–92.

Van Leeuwen, D., Brummer, N., 2006. Channel-dependent GMM and multi-class logistic regression models for language recognition. In: Proceedings of the IEEE Odyssey – The Speaker and Language Recognition Workshop, pp. 1–8.

Veselý, K., Karafiát, M., Grézl, F., 2011. Convolutive bottleneck network features for LVCSR. In: Proceedings of the Automatic Speech Recognition and Understanding, pp. 42–47.

Veselý, K., Karafiát, M., Grézl, F., Janda, M., Egorova, E., 2012. The language-independent bottleneck features. In: Proceedings of the IEEE Workshop on Spoken Language Technology, pp. 336–341.

Welling, L., Kanthak, S., Ney, H., 1999. Improved methods for vocal tract normalization. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, vol. 2, pp. 761–764.

Zazo, R., Lozano-Diez, A., Gonzalez-Dominguez, J., Toledano, D., Gonzalez-Rodriguez, J., 2016. Language identification in short utterances using long short-term memory (LSTM) recurrent neural networks. PloS ONE 11, 1.

Zissman, M., Singer, E., 1994. Automatic language identification of telephone speech messages using phoneme recognition and N-gram modeling. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, vol. I, pp. I/305–I/308.